Project Schedule Simulation: Incorporating Human Factors' Uncertainty and Features' Priority in Task Modeling

Mizouni, Rabeb; Lazarova-Molnar, Sanja

Published in:
Journal of Software

Publication date:
2015

Citation for pulished version (APA):
Mizouni, R., & Lazarova-Molnar, S. (2015). Project Schedule Simulation: Incorporating Human Factors' Uncertainty and Features' Priority in Task Modeling. Journal of Software, 10(8), 939.

Project Schedule Simulation: Incorporating Human Factors' Uncertainty and Features' Priority in Task Modeling

Rabeb Mizouni[1*], Sanja Lazarova-Molnar[2]
[1]Department of Electrical and Computer Engineering, Khalifa University, UAE.
[2]Centre for Smart Energy Solutions, University of Southern Denmark, Denmark.

* Corresponding author. Email: rabeb.mizouni@kustar.ac.ae
Manuscript submitted November 27, 2014; accepted June 25, 2015.
doi: 10.17706/jsw.10.8.939-960

Abstract: Despite several attempts to accurately predict duration and cost of software projects, initial plans still do not reflect real-life situations. Since commitments with customers are usually decided based on these initial plans, software companies frequently fail to deliver on time and many projects overrun both their budget and time. To improve the quality of initial project plans, we show in this paper the importance of (1) reflecting features' priorities/risk in task schedules and (2) considering uncertainties related to human factors in plan schedules.

To make simulation tasks reflect features' priority as well as multimodal team allocation, enhanced project schedules (EPS), where remedial actions scenarios (RAS) are added, were introduced. They reflect potential schedule modifications in case of uncertainties and promote a dynamic sequencing of involved tasks rather than the static conventional way of modeling schedules. This paper describes a methodology to introduce EPS models in the development of project plans. Our case study findings show clearly the ability of the EPS model to anticipate delays due to human factors and to predict remedial scenarios to adjust to such delays. By comparing EPS to classical schedules, EPS simulation provides more accurate results with regards to project goals.

These instructions give you guidelines for preparing papers for Journal of Software (JSW). Use this document as a template if you are using Microsoft Word 6.0 or later. Otherwise, use this document as an instruction set. The electronic file of your paper will be formatted further at Journal of Software. Define all symbols used in the abstract. Do not cite references in the abstract. Do not delete the blank line immediately above the abstract; it sets the footnote at the bottom of this column.

Key words: Human resources allocation uncertainty, feature priority/risk, dynamic project schedule, simulation.

1. Introduction

Project management is the discipline of planning, organizing, and managing resources to achieve specific project goals and objectives. It is the process that uses schedules to plan and subsequently report progress within project's environment. With the increasing size of today's software project, it is well recognized that the success of these fairly large projects is directly related to their *management* [1]. Not surprisingly, many researchers have identified poor planning as one of the primary causes for failure of software projects [2]-[4]. A common practice in software development is to first establish an *initial software project plan*. It is

usually concerned with defining project mission, developing project schedule plans, client consultation and client acceptance [3]. Moreover, it is used as a basis on which to make delivery commitments to clients; hence the importance of constructing a credible plan that provides an accurate estimation of completion dates.

Even though planning is an important step in most of the software development processes, and even though it is performed in most of the large software projects, an effective planning is rarely accomplished because many interrelated factors increase the uncertainty that a project is surrounded by. These factors include task durations and resources sharing. Very often an activity in real industrial software project takes more than the originally estimated duration, leading to a delay in the remaining activities due to resource unavailability [5]. Consequently, many projects overrun their budget and time.

Recently, the focus was shifted to the quality of initial project plans. Many attempts have been conducted to improve the design of initial project schedules [6]-[10]. On one hand, they endeavor to calculate an optimized schedule for a given project, but on the other hand many of them use over-simplified models that underestimate the interrelated nature of many project factors, affecting badly the accuracy of the resulting plan. Uncertainty in human resource allocation and availability is typically one of the factors that is frequently over-simplified and overlooked. While it has been defined over and over (Refer to [11]), uncertainty can be seen as any issue (aspect) in a project and its environment that can arise during its execution and may require a fast and rapid adaption of the plan to cope with it. Uncertainty can affect many other project factors such as task durations, feature modifications, and resources' unavailability. In this paper we are concerned with human resources allocation uncertainty and its effective modeling within a project plan to improve the schedule's quality.

1.1. Problem Statement

As in any project, software project managers deal with people, machines, and materials. These resources usually have limited availability. A common practice is to define an initial plan where efficient allocation of the available resources to the various project activities is considered. Despite this initial allocation, software project plans are frequently prone to changes. Unexpected human-related factors may eventually emerge during the project execution such as illness of team members, unforeseen implementation problems that requires more human resources on the activity, a team member turnover, etc... As a result, an activity in a real-life industrial project can take more than its original duration estimate, leading to a delay in the rest of the activities due to human-resources' unavailability and/or reallocation. In such situation, the initial project plan is re-examined to optimize resources usage and answer these new features to avoid any delay in the product delivery date. Intuitively, we can think that reallocating teams will improve the project plan. However, this is not as simple as it looks. Let us analyze this situation more comprehensively.

In a classic scenario, during project run, a manager may face a situation where a team that was originally assigned to the implementation of a feature is unavailable while at the same time another team is available. In such situation, the manager has three choices:

1) Either to assign the activity "in hold" to the available team: As we may expect, teams have different levels of expertise, a fact that will in turn affect task's duration. The available team may need longer time to achieve the task than the initially assigned team. In addition, the manager needs to make a decision concerning the team that was originally assigned to the task. Once the team becomes available, it can start the next assigned task, it can consider working on a task that was not initially part of its responsibility, or it can simply wait. Re-scheduling of tasks, as well as re-allocation of human resources, is possibly required in this case.
2) To stick to the initial plan and wait for the assigned team: The waiting time of teams will increase and any delay in the execution of a task will potentially cause the delay of the whole project delivery. With

nowadays-evident pressure of time-to-market, this solution cannot be adopted at all times.

3) Cancel the implementation of the task: In an extreme case, the project manager can decide to cancel the implementation of the delayed task if he thinks that it will not affect the customer satisfaction or because of an extreme delivery pressure. Again, this solution can only be adopted in certain situations.

To cope with these types of scenarios, it is important to provide managers with simulation approaches and tools that are flexible enough to handle resources' uncertainty that surrounds software projects. These approaches are supposed to explore various possible scenarios that manager's decisions may cause and improve the decisions about task's rescheduling and its subsequent resources' reallocations.

1.2. Proposed Solution

In a software project, once the features are defined and selected, they are assigned a priority level ranging from "must-have" (MH) to "nice-to–have" (NTH). This prioritization helps the project manager to resolve conflicts and, more importantly, to perform trade-offs throughout the development lifecycle [12]. Usually, a common practice in case of resources insufficiency is to first implement the features that have higher levels of priority. The implementation of the features with lower priority may be postponed, or even cancelled, according to the time-to-market and cost of the project.

To improve the accuracy of project plans, it is required that schedules incorporate both resource allocation uncertainty and features' priorities. For this purpose, we proposed in [13], to consider multi-modal tasks, where an activity can be processed in one of several modes, each defining a different options in terms of human resources allocation and duration. In addition, we extended the classic project schedule model to define several types of tasks to support flexibility in feature scheduling and, thus, cope with resources unavailability. This flexibility is based on the priority of the feature the task is implementing. Analogous to features priorities, we defined two types of tasks cancellable and non-cancellable.

In this paper, we extend the model to have tasks that range from "non-floating" and "non-cancellable" to "floating" and "cancellable" tasks. Floating task is a task that anticipates high uncertainty and is highly flexible in terms of its human resource allocation. It has the property of being flexible in its order of execution with respect to other tasks. Cancellable task is defined as a task that can be cancelled without harming the project success. Rather than a static task sequencing, floating and cancellable task introduce dynamicity to project. A mapping between project features to tasks is achieved based on features priorities and associated implementation risk factor. We also extend the definition of *Enhanced Project Schedule* with floating and cancellable tasks. Simulation of the proposed schedule is used to allow analyzing potential effects that uncertainties in project schedules may cause on meeting project goals and, thus, computing their significance in a given project.

In this paper, we define a methodology to use efficiently EPS in project management. This methodology relies on an iterative process that takes advantage of the flexibility EPS is offering to simulate many redial action scenarios and assess their output with respect to a simulation goal set by the manager. By comparing different scenarios, managers can make their decision on the project initial plans based on sound results rather than their personal experience. Finally, we present a case study where we compare between the simulation results of the EPS, a classical schedule, and a model that do not consider task cancelation. We demonstrate that the EPS is in fact introducing more accuracy and can help to predict in an efficient way the set of tasks that will be affected by delays and human uncertainty factors.

To summarize, the contribution of this paper is threefold:

1) Extend and improve the model presented in [13] to provide a more efficient mapping between project features and simulation tasks
2) Define a methodology to use EPS in software project management and cope with requirements change related to human factors or task cancelation.

3) Compare the simulation results of the EPS and classical schedules to assess the advantages/disadvantages of EPS.

1.3. Paper Organization

The remainder of this paper is structured as follows: In the next section we provide an overview of the importance of effective project planning in the software industry. Then, we summarize the related work on coping with uncertainties in project scheduling. Section 3 we present the EPS model. Section 4 proposes a methodology to incorporate the use of EPS model in project management and plan. Section 5 and 6 illustrate our approach through the simulation of the HOme Lightening automation System (HOLIS) case study as well as the results of the simulation. Finally, Section 7 concludes the paper and overviews some future directions.

2. Preliminaries: Software Project Planning

Before presenting our solution to incorporate human resources allocation uncertainty in project scheduling, we will outline some preliminaries about project planning. In this section, we sketch the key issues of features management and human resource allocation in project planning.

2.1. Features Management in Project Planning

A software process is a set of activities, methods, practices, and transformations that people use to develop and maintain the software and its associated products [14]. A project consists of a number of tasks (activities) where a predefined set of tasks has to be processed in order to complete the project. The tasks are interrelated by two constraints:

Precedence constraints: tasks cannot be undertaken in any order and some tasks cannot start unless others have been already completed. This order is usually defined when specifying the various features to implement, and the precedence is not subject to change even in case of plan re-scheduling. Consequently, if the implementation of the feature F_1 precedes the implementation of the feature F_2, any change in the project plan has to verify this condition.

Resource sharing: task execution requires efficient resources' management. Such resources may include financial resources, inventory, human skills, production resources, information technology (IT), etc.

It is common in real-life software project to have hundreds of distinct features. However, due to limited resources that in terms of staff, budget and time, it is highly critical to *select* and *prioritize* features. The selection and prioritization are usually based on each feature's importance, time and effort, as well as the resources needed vs. their availability to implement it.

The author in [15] proposes two means of feature's prioritization: (1) by implementation order, which is the implementation order of the features in an incremental and iterative development cycle, and (2) by importance, which corresponds to the order by importance to some stakeholders (business value, cost of implementation, and risk). In this paper, we adopt the second approach, i.e. we prioritize features by their importance. The priority of a feature will be reflected in the schedule task nature, as we will present in the next section.

Different scales of priority are used to classify features. We use a typical three level scale, termed as: *critical*, *important*, and *useful* features [16]. *Critical feature* is a must-have feature that needs to be delivered on time and the success of the project depends on it. An *important* feature is a feature that has to be delivered by the end of the project but its implementation can be delayed if needed. A *useful* feature is a nice to have feature that will be implemented only if resources are available, i.e. time, budget, and human resources.

2.2. Human Resources Allocation in Project Management

Workforce allocation is seen as an important phase in software project management. In this phase, software developers are being assigned to the various project tasks based on the software development process they are taken into consideration [17]. While an initial allocation of software designers/developers can be calculated based on the initial set of features, workforce adjustments during project performance often become necessary for several reasons:

1) Projections recalculation, based on workforce size variability, and current development productivity [17], often project activities are either cancelled or implemented by resources that are not exclusively reserved for the current project [18]. Changes of either team members or task assignments may occur, introducing changes in the project baseline.
2) Number of remaining features to be implemented and the time left with respect to the software delivery date. In many cases, when the deadline is approaching a re-allocation of the human resources is made to increase the chances of delivering critical and important features in the established deadlines.
3) Requirements volatility [19], known as one of the main causes of software project failure. Despite advances in software engineering over the last decades, most software projects still experience considerable requirements' changes during their development as a consequence of customers' required changes, initial requirement misunderstanding, technology change, or even political changes. Requirements volatility refers to the growth or changes in project functional or non-functional [20] requirements during development lifecycle either by deletion, addition or modification [21]. Such changes usually impact the set of features under development. While volatility is expected and managed in the requirements engineering phase, it dramatically impacts the project schedule when changes occur in the later stages of software development lifecycle.

Because of the above-mentioned reasons, it may happen during re-scheduling that a team is assigned to a task that was originally matched to another team during the workforce planning. Such scenario is not easy to predict during project scheduling because it is difficult to forecast when such an adjustment may happen. Building a robust schedule that considers this uncertainty factor is critical to software projects success. Such schedule needs to forecast possible deviation from the originally planned schedule and allow and support the adjustment in such situations with minimal waste in terms of delays and resources waiting time.

The incorporation of uncertainty in project planning and scheduling has resulted in numerous research efforts, particularly focusing on uncertainty in task duration or cost [8] as we present next.

2.3. Related Work: Uncertainty and Scheduling

Despite the fact that research on project scheduling has widely expanded over the last few decades, the majority of the research efforts are focused on building a schedule that assumes complete information and a static deterministic problem environment, which has been shown to be a non-realistic assumption [22]. Such a resulting schedule can only be considered as a baseline for the execution of the project. Often, during project execution, the baseline schedule is subject to uncertainties, leading to schedule changes, thus increasing the need to consider uncertainty in project scheduling.

According to the literature, uncertainty in project scheduling has been considered in a number of ways [8], [11], [23], [24]. The authors of [8] have distinguished five approaches to deal with uncertainty :

Reactive scheduling [25]-[27] which ignores uncertainty in creating the baseline schedule and revises or re-optimizes it when an unexpected event occurs. It is based on repairing actions, and spans from the definition of basic rules such as the right left shift rule to a full rescheduling pass the remaining part of the project under consideration.

Stochastic scheduling which considers uncertainty in activity durations. Importance is given to stochastic resource-constrained project scheduling. It mainly focuses on minimizing the expected project duration, subject to precedence constraints and renewable resource constraints. It is defined as "A set of jobs with random processing times that have to be executed subject to both precedence and resource constraints" [28]. The authors of [18] focused on modeling stochastic resources availability. They propose a mathematical model and suggest an algorithm for building a more robust schedule and repairing the disruption due to resource unavailability. In our proposed approach, we extend this assumption and go even further by modeling various types of resource changes. This includes re-allocation of teams because of the unavailability of resources, or any other internal team change such as member's illness and/or expertise modification.

Scheduling under fuzziness, as first applied by Prade [29]; it uses fuzzy variables to describe duration times via expert knowledge rather than using probability distributions. Fuzzy theory is seen as an approach to adapt scheduling models into reality [30]. Recent research projects in this field present methodologies that calculate the fuzzy completion project time Feng-Tse, [31] or obtain fuzzy critical paths and critical activities and activity delay [32].

Proactive (robust) scheduling, as defined in [24] *proactive project scheduling deals with uncertainty by creating a baseline schedule that is as much as possible protected against disruptions*, assuring a certain robustness in the resulting project scheduling. Herrolem and Leus [33] examined various procedures for development of a stable pre-schedule, which is unlikely to undergo major changes when it needs to be repaired when activity duration changes. They proposed a mathematical model to minimize the expected weighted deviation in activity start times. However, they made abstraction of resources usage assuming a proper allocation of resources has been performed.

Sensitivity analysis is studying the impact of uncertainty on the output of the scheduling algorithms. In [34], the authors study the degradation of the performance of the solution due to the disturbances by analyzing the impact of the perturbations on the computed schedule.

In all of the abovementioned approaches, the evolution structure of the precedence is deterministic. In the research work we present in this paper, we deal with this aspect in a more realistic way by considering the fact that some precedence relationships may change under some conditions. In addition, stochastic resource-constrained project scheduling generalizes the so called *stochastic project networks* or *PERT-networks* job processing times that presume unlimited availability of resources. In our approach, we assume that we have limited resources availability, which is usually the case in real-life software projects.

One of the used tools for project management analysis is software process simulation and modeling [35]. Simulation is a technique that has been successfully applied in many domains [36-38] . In project scheduling, it can be used to forecast the effort/cost, schedule and the product quality, the staffing levels as well as the expertise needed across the time according to the task precedence constraints, the estimation of the resource constraints and hence better resource allocation; and finally an efficient analysis of risks.

3. State-Dependent Tasks

Recall that the objective of this work is to add robustness to the baseline project schedule by introducing human uncertainty. In our case, we are interested in interrelating human resource allocation uncertainty and features priority levels in the simulation model to obtain a robust baseline.

To formalize uncertainty in human resources allocation with respect to features priorities, we define the notion of highly uncertain state-dependent task, for which we allow all relevant parameters to determine its duration. Before we introduce our simulation model, we first outline a few necessary assumptions upon which we build our project schedule.

3.1. Assumptions

We assume that:

1) The manager is given a pool of human resources with different degrees of productivity to perform different types of tasks.
2) Each team is responsible for completing the implementation of the task as a whole.
3) Change in the initial structure of the team is equivalent to a change of the team. In other words, if one member of team A is absent for illness, or has to be transferred to another team to serve another task, then the resulting team is in fact team B that demonstrates a new expertise. In this particular case, the duration needed for team B to implement a task will be longer/shorter than the duration needed for team A.
4) Tasks are sharing human resources and their implementations obey to a certain precedence order.
5) Tasks are non-interruptible activities. Once the implementation of a task starts, it is not interrupted.
6) Duration of a task is modeled by a probability distribution function.
7) For activities that may be implemented by more than one team, duration probability distributions of each team need to be specified.
8) Features are correctly prioritized. Change of prioritization [15], as it may happen during project implementation, due to addition/removal/change of other features leads to a change in the EPS as we will describe in the methodology proposed.
9) Type of a task depends on the priority of the feature that the task is implementing and the implementation risk associated to it.
10) When developing the state space of a schedule, the highest priority features have to be implemented first as part of the scheduling when possible.

3.2. Modeling and Classification of Features

We propose to distinguish tasks in nature so they reflect priorities of the features they are implementing to introduce dynamic scheduling, on one hand, and flexibility in human resources allocation, on the other hand. We distinguish four types of tasks, identified as: *non-floating non-cancelable* tasks, *floating non-cancelable* tasks, *floating cancelable* tasks and finally *non-floating cancelable* tasks. They are defined as follows:

Non-floating and non-cancellable (NFNC) tasks: They typically implement features that are estimated as critical for the success of the project. These tasks do not have any flexibility in terms of human resources allocation. They are assigned only to experienced professionals to reduce the risk of their failure. Consequently, resources assignment of **NFNC** tasks follows a fixed resource allocation strategy which implies a single team responsible for their implementation. These tasks cannot be cancelled.

Floating and non-cancellable (FNC) tasks: these tasks have flexibility in terms of human resource allocation. They usually implement features that are not associated with high risk. Resource allocation for floating and non-cancellable tasks can follow a multi-modal strategy, where many teams can implement them with respect to their availability. During project's lifecycle, any of the teams that become available can implement it in order to optimize project's duration and maximize resource utilization, in accordance with project goals. In general, FNC tasks invite various on-the-fly decision scenarios since they are executed when the resources are available because they allow certain flexibility in team's allocation. As consequence to human resources flexibility, FNC tasks can be postponed and rescheduled in the project plan however they can never be cancelled.

Floating and cancelable (FC) tasks: These tasks have flexibility in terms of human resource allocation, yet can be cancelled if needed. In fact, when teams responsible for implementing the task are either unavailable or solicited to do more important tasks then the task is postponed. If the project is overdue,

then these tasks are cancelled. The resource allocation for **FC** tasks follows a multi-modal strategy, where many teams can implement them with respect to the team availability.

Non-floating and cancelable (NFC) tasks: These tasks have no flexibility in terms of human resource allocation and yet can be cancelled if necessary. Resource allocation for **NFC** tasks follows fixed strategy, where only one team can implement them. If the team is not available, the implementation of the feature is cancelled. Consequently, **N**FC tasks have a very low probability of getting implemented compared to other types.

Table 1 presents a possible mapping between features priority levels and their possible assigned task. Let us outline some facts:

1) As expected, critical and important features can never be modeled as cancelable tasks. Only nice-to have features can be cancelled.
2) While it is possible to model useful features as NC tasks, we do not recommend it. It prevents the model from certain flexibility in the task execution order, a decision that may delay the overall duration of the project.
3) Critical and important features have to be modeled as non-cancellable tasks, either floating or non-floating. Such decision depends on the risk level the implementation of the feature presents. When the risk of the feature is high, it would be more judicious to assign its implementation to an experienced team and hence model it as NFNC. However, when the feature is critical but presents low risk and the project manager has some flexibility with regards to the delivery date of the feature, then s/he may tolerate model it as FNC.

Let us now formally define a task. We extended the definition in [39] to incorporate the different type of tasks.

Let R= {R_i, 1≤i≤n} be the set of n features of the software to implement. Let T= {T_i, 1≤i≤m} be the set of m teams (representing the human resources available for that software project).

Definition 1: Task

A task is a 3-tuple task =(R_i, D, *type*) where :

1) $R_i \in R$ the feature the task is implementing.
2) type ∈{**NFNC, FNC,NFC, FC**} the type of the task
3) $D = \{D_{ij} (R_i, T_j)/T_j \in T\}$ is the set of predetermined task durations expressed as functions of the capable teams T_j and feature R_i.

For **NFNC** and **NFC** tasks, D is a singleton.

$$D = \{d_{ij}(R_i, T_j), R_i \in R, T_j \in T\}$$

Table 1. Mapping Feature Priority Levels to Task Types

Task		NFNC	FNC	NFC	FC
Critical	High Risk	X			
	Low Risk		X		
Important	High Risk	X			
	Low Risk		X		
Useful	High Risk	X (not recommended)	X (not recommended)	X	
	Low Risk	X (not recommended)	X (not recommended)		X

Contrarily to classical schedules, the task has a type. This type reflects the priority of the feature the task

is implementing. In addition, it may be defined as a multi-modal task that can be implemented by many teams adding as such flexibility to fixed human allocation usually adopted when defining projects' schedules. These new task descriptions will be used to define the *Enhanced Project Schedule*.

3.3. Enhanced Project Schedule

The project schedule model needs to reflect the nature of tasks in the simulation itself. To achieve this goal, we propose to extend the project schedule with *fuzzy rules*. Fuzzy rules are conditional statements that express potential deviations to the initial project plan and remedial scenarios that should be undertaken consequently. Each fuzzy rule is made up of two parts: condition and action, formally written as "condition ⟹ action". Conditions can be described either by using strict terms, or fuzzy ones. The action can typically be canceling or interrupting some of the tasks, or one of the various types of rescheduling. This fact makes our schedule description evolving, rather than rigid and inflexible. An example for a fuzzy rule would be:

$$Task_x\ takes\ too\ long \Rightarrow cancel\ Task_y$$

or

$$Task_x\ completes\ quickly\ after\ Task_y \Rightarrow cancel\ Task_z$$

Both are examples for typical proceedings during project execution. However, in our approach we allow for their modeling, assessment and quantitative evaluation.

Fuzzy rules support the conversion of the type of the task in the project schedule model. Hence, these rules should be consistent with the type of the task specified by the analyst. As an example, a rule can never recommend to cancel a NFNC and FNC tasks as they represent *critica*l and *important* features. Consequently, the two rules mentioned-above are correct only if $Task_z$ and $Task_y$ are cancellable tasks.

Definition 2: Enhanced Project Schedule

An Enhanced Project Schedule (EPS) is a 5-tuple $EPS = (Tasks, P, T, F, IGC)$ where:

1) Tasks $= \{Task_1, Task_2, \dots, Task_n\}$, set of tasks, where each task corresponds to a task in the project schedule
2) $P = \{P_1, P_2, \dots, P_m\}$, where $P \subseteq Tasks \times Tasks$ *is* the set of precedence constraints, that are actually tuples of two tasks where the completion of the first one is a pre-feature for commencing the second one, e.g. $(Task_x, Task_y)$ would mean that completing of $Task_x$ is a pre-condition for beginning $Task_y$
3) $T = \{T_1, T_2, \dots, T_l\}$, set of teams available for the execution of the project
4) $F = \{f_1, f_2, \dots, f_s\}$, set of fuzzy rules that define remedial action scenarios.
5) IGC − Initial Gantt Chart, initial sequencing of tasks that satisfies the set of precedence constraints provided by P.

Note that *F* can be an empty set too, which would imply sticking to the original project schedule provided by IGC. Fuzzy rules ensure the dynamicity of the schedule.

3.4. Application to the Proxel-Based Simulation

To implement our model, we need to extend simulation approaches to cope with the dynamic task scheduling. In [13], we proposed to extend the proxel based simulation. The proxel-based method, first introduced in [40], [41], is a relatively novel simulation method, whose underlying stochastic process is a discrete-time Markov chain [42] and implements the method of supplementary variables [43]. The method, however, is not limited to Markovian models. On the opposite, it allows for a general class of stochastic models to be analyzed regardless of the involved probability distribution functions. In other words, the proxel-based method combines the accuracy of numerical methods with the modeling power of

discrete-event simulation.

The proxel-based method is based on expanding the definition of a state by including additional parameters which trace the relevant quantities in one model through a previously chosen time step. Typically this includes, but is not limited to, age intensities of the relevant transitions. The expansion implies that all parameters pertinent for calculating probabilities for future development of a model are identified and included in its state definition.

Proxels, as basic computational units of the algorithm, follow dynamically all possible expansions of one model. The state-space of the model is built on-the-fly, as illustrated in Fig. 1, by observing every possible transiting state and assigning a probability value to it (*Pr* in the figure stands for the probability value of the proxel). Basically, the state space is built by observing all possible options of what can happen at the next time step. The first option is for the model to transit to another discrete state in the next time step, according to the associated transitions. The second option is that the model stays in the same discrete state, which results in a new proxel too. Zero-probability states are not stored and, as a result, no further investigated. This implies that only the truly reachable (i.e. tangible) states of the model are stored and consequently expanded. At the end of a proxel-based simulation run, a transient solution is obtained which outlines the probability of every state at every point in time, as discretized through the chosen size of the time step. It is important to notice that one source of error of the proxel-based method comes from the assumption that the model makes at most one state change within one time step. This error is elaborated in [41].

Each proxel carries the probability of the state that it describes. Probabilities are calculated using the instantaneous rate function (IRF), also known as hazard rate function. The IRF approximates the probability that an event will happen within a predetermined elementary time step, given that it has been pending for a certain amount of time Δt (indicated as 'age intensity'). It is calculated from the probability density function (f) and the cumulative distribution function (F) using the following formula:

$$\mu(\Delta t) = \frac{f(\tau)}{1 - F(\tau)} \tag{1}$$

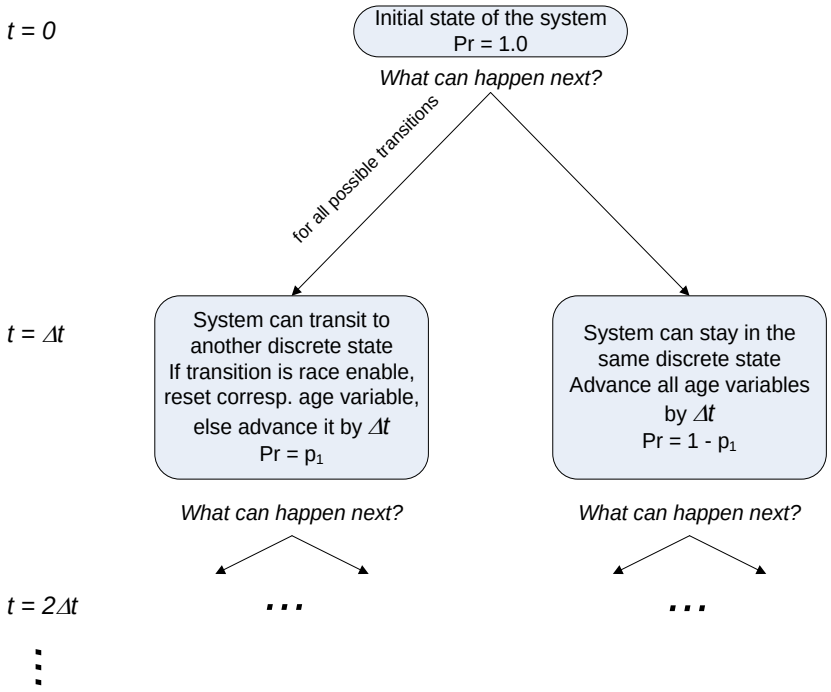

Fig. 1. Illustration of the development of the proxel-based simulation algorithm.

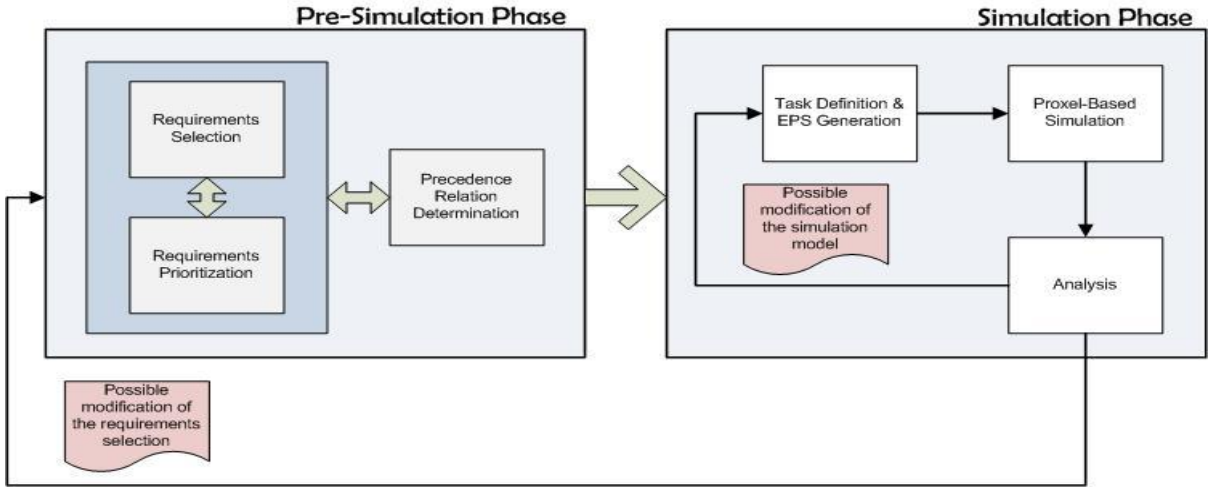

Fig. 2. Simulation methodology using the enhanced project schedule.

For our purpose we extended the original proxel-based simulation algorithm to account for the fuzzy scenarios. They fitted straightforwardly into the existing framework. In addition, the algorithm was adapted to collect statistics about the probability of having a certain feature implemented. More on the extension of the proxel based simulation can be found in [13].

As all state-space based methods, this method also suffers from the state-space explosion problem [44], but it can be predicted and controlled by calculating the lifetime of discrete states in the model. In addition, its efficiency and accuracy can be further improved by employing discrete phases and extrapolation of solutions [45]. More on the proxel-based method can be found in [41].

4. A Methodology for Using EPS in Simulating Software Project Plans

As Shown in Fig. 1, we define a methodology for project scheduling using the EPS model. This methodology relies on two main phases: a *pre-simulation* phase and a *simulation* phase.

4.1. Pre-simulation Phase

Often, there are more features specified by the customer than can be implemented with the actual project resources. As mentioned previously, the project manager, through several negotiations with the customer, have to select and prioritize these features. At the end of this step, it is expected that the features to be implemented and delivered are chosen and accepted by the customer. Obviously, this step is challenging. The selected features have to satisfy the interests of the various stakeholders involved in the project and take into considerations the importance of business value, the risk and cost of development, and finally features dependencies [46]. After several iterations of selecting, prioritizing the features, and defining the precedence order between them, the pre-simulation phase ends by delivering a set of features, their respective priorities as well as the precedence order to respect during their implementation. Normally, not all features are coupled with a precedence order. Our approach takes advantages of this fact to add flexibility to the schedule under simulation.

4.2. Simulation Phase

It consists of three steps:

1) Generation of EPS: which is threefold: (1) selecting the teams that have the potential to implement the features and estimating their level of expertise on the one hand, and attributing a probability function that will be used during the simulation on the other hand. Defining probability functions rely on previous data available to the analyst describing duration of time needed to implement a feature in similar situation with a comparable team expertise. (2) Define an initial Grant chart with respect to

precedence order of features as defined in the pre-simulation phase. The sequencing of the tasks in the initial Gantt chart gives the starting state of the simulation from where the different possibilities will be investigated (3) Generation of fuzzy rules in accordance to task types.

2) Proxel-based simulation step: The generated EPS in the previous step is run by our proxel-based simulator or any other simulation tool that has been extended to process EPSs.
3) Analysis step: The manager has now to interpret the obtained simulation results. We believe that a refinement of features at this level may be considered if the project runs out of the target time and/or budget specified during the simulation.

Using the simulation model that we defined, the manager will have the necessary framework to better select the different features. An important aspect is the feedback that our approach is offering to him. In addition, using our model, the manager will be able to detect the idle time of the different teams. This information is very useful in initial project planning. It can be used to improve it either by refining the feature selection by adding or deleting some nice to have features, or by re-allocating human resources if the waiting time of certain teams is relatively long. Hence, our model will help the efficient allocation of team in order to achieve an effective time management.

Finally, our model can be used in an iterative process, where the manager will first consider all the features defined by the customer. Then, by simulating our EPS model, he will refine the set of features so that the project fits well within the human resources and time assigned to it.

5. Proof of Concept

HOLIS is a home lightening automation system that brings new lightening automation functionality with ease of use, comfort, and safety. It is intended to be used by homeowners' buildings and high-end homes. Details of the case study can be found in [16]. To simulate the schedule of the development of HOLIS, we select a subset of the different features that the system should implement. We follow the methodology we presented in Fig. 2 to present our case study.

5.1. Pre-simulation Phase

Table 2 summarizes the subset of the HOLIS system features with their respective priorities and the effort needed to implement each of them. Effort has one of the three typical levels: low, medium, and high. Unsurprisingly, the effort needed to implement a task is team dependent. The values given in Table 2 represent an estimation of the effort needed to a **trained** team to implement a given feature.

The third column represents the precedence constraint to respect when implementing the different features. *Null* indicates that the implementation of a feature is independent from the others. For instance, **Feature1** can be implemented without any constraints while the implementation of **Feature2** cannot start unless **Feature1** has been already implemented. Fig. 3 shows the precedence relationship between the different features of the HOLIS case study. Any proposed schedule has to comply with it.

5.1.1. Simulation phase

According to our methodology, after defining the different features, their priorities and the precedence relationship between them, an EPS can be defined. At this level, a mapping between each feature and its corresponding task in the simulation schedule has to be defined.

Let us consider our running example. We propose to categorize the set of prioritized features as **NFNC**, **FNC**, **NFC** and **FC** tasks. Fig. 3 shows this mapping. The six first features are all mapped to non-cancelable tasks, either floating or not. It is mainly because of their priorities (Critical and Important). **Feature4** and **Feature5** are considered as **FNC** allowing certain flexibility in their implementation**. Feature7** is mapped to **FC** task while **Feature8** is a **NFC** task**.**

In order to simulate the HOLIS system, we process to the human resource allocation. We suppose that we

have two teams working on the HOLIS project: *Team A* and *Team B*. It is obvious that these two teams have different expertise, and each will fit better to a particular task. As explained previously, any software project is subject to workforce adjustment. During this case study, we anticipate such on-the-fly decision and allow the expression of possible decisions, while simulating the project duration. Table 2 illustrates the task allocation for each of the teams. It also shows the estimated effort that the assigned team needs to implement a feature. All the non-floating tasks are assigned to a single team and this team is not subject to change even if an uncertainty arises.

Table 2. Features of the HOLIS System

Features	Priority	Risk	Effort	Preceence
Feature 1: automatic timing settings for lights and so on.	Critical	High	Low	Null
Feature 2: built-in security features (alarm, bells)	Critical	High	Medium	Feature1
Feature 3: non-PC control unit	Critical	High	High	Null
Feature 4: vacation settings	Critical	Low	Low	Feature3
Feature 5: uses my own PC for programming	Important	Low	High	Null
Feature 6: close garage doors	Important	High	Low	Feature 1
Feature 7: automatically turn on closed lights when door opened	Useful	Low	Low	Feature6
Feature 8: Interface to audio/video system	Useful	High	Medium	Null

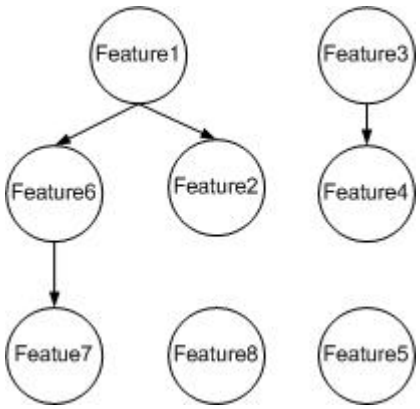

Fig. 3. Features precedence relationship.

Finally, we draw the attention to two useful aspects our EPS may help the manager in:

1) Verification of the consistency of the assigned features priority. It is obvious that a FC-task cannot have a precedence constraint with a **NFNC** task since it is against the semantic of the floating task as well the implementation flexibility this task is offering;
2) According to the precedence constraint of the different features of the HOLIS system and their respective priorities, only one starting point to our project schedule can be defined: *Team A* implementing **Feature1** and *Team B* implementing **Feature3**. However, in other projects, we may have several starting points. The project manager needs to investigate these possible starting points in order to identify the best that serves the project scheduling objectives.

At this point, the manager is able to generate an EPS to be used as a simulation model. He needs to use the precedence graph shown in Fig. 3 as well as teams and task type allocation in Table 3. The simulation model is shown in Fig. 4. This model clearly states that Feature 4, 5, and 7 can be implemented by any of the available teams, to the opposite to the remaining features.

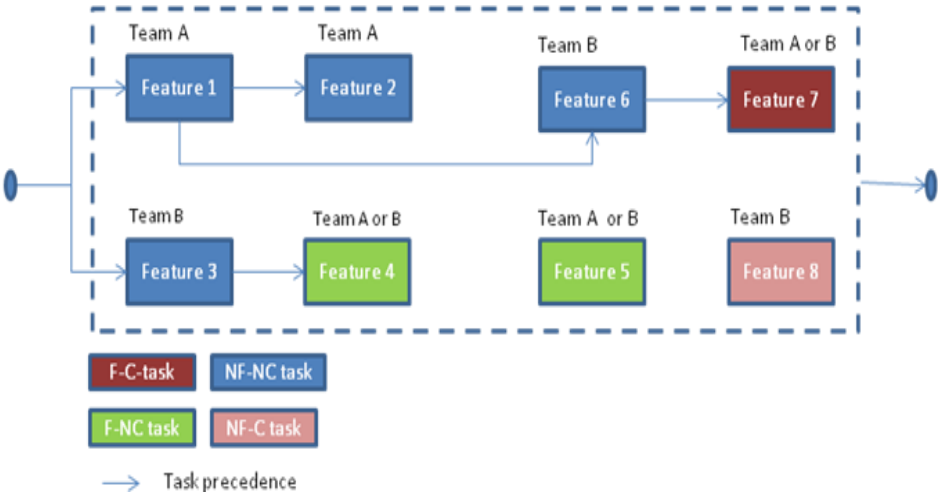

Fig. 4. EPS generation for the HOLIS system.

6. Experiments and Results

The experiments were run on a standard workstation with an Intel Core2Duo Processor at 2.0 GHz and 1 GB RAM. The choice for Δt was 0.1 and the simulation was run up to time t = 20. This implies that the number of simulation steps was 200. The computation time for this experiment was ca. 4 seconds. In the following we present the results, i.e. the statistics that were calculated during this simulation experiment.

In order to see the difference our new task modeling makes, we propose to have three different simulations and compare among them.

6.1. Conventional Simulation Following a Gantt Chart (Case A)

In the current simulation approaches, uncertainty, as well as flexibility, in human allocation is hardly represented. Table 3 shows a possible distribution of HOLIS features into simulation tasks in the case of conventional simulation approach.

Table 3. HOLIS Conventional Distribution of Tasks

Features	Effort (for trained team)	Assigned Team and Estimation of the Needed Effort
Feature 1	Low	*Team A* (Effort = Low)
Feature 2	Medium	*Team A* (Effort = Medium)
Feature 3	High	*Team B* (Effort = High)
Feature 4	Low	*Team B* (Effort = Low)
Feature 5	Medium	*Team A* (Effort = Medium)
Feature 6	Low	*Team B* (Effort = Low)
Feature 7	Low	*Team A* (Effort = Low)
Feature 8	Medium	*Team B* (Effort= Medium)

Tasks of the HOLIS system can be represented using this simulation scheduling in Fig. 5.

The state space of this rigid schedule is shown in Fig. 6. Twenty discrete states are generated and the sequencing denoted by the Gantt chart is respected. Neither the priority nor the human resource allocation uncertainty is taken into consideration.

6.2. Simulation with Priority Consideration but without Fuzzy Rules (Case B)

In this case, we simulate the model by taking into consideration the priority of the feature while developing the state space of the HOLIS system as well as the human resource flexibility. However, we do not allow the cancelation of tasks. Consequently, **Feature7** and **Feature8** are modeled as NFNC and FNC

tasks respectively.

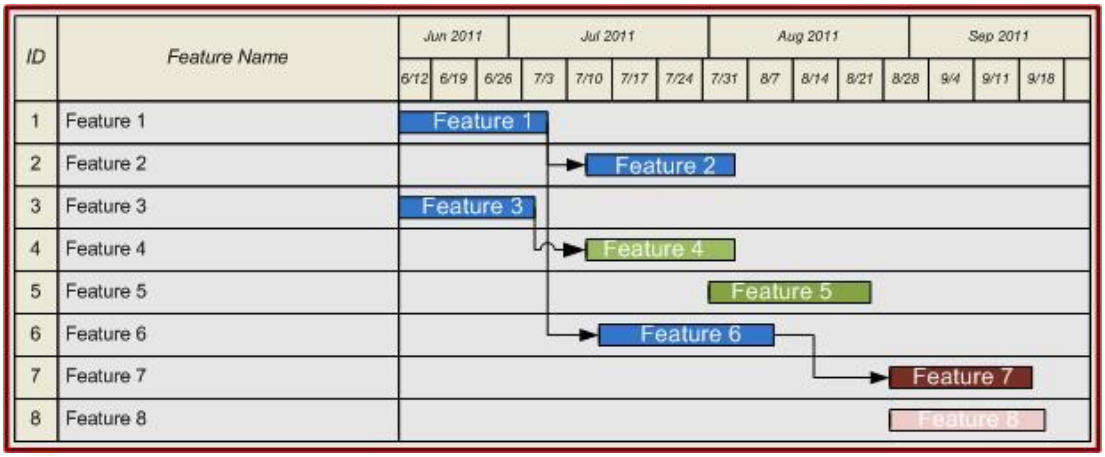

Fig. 5. Gantt chart for the HOLIS system.

In the following we provide the details of the proxel-based simulation of the sample project schedule that involves the different tasks we are proposing. This should serve as description of our approach through an example.

Each task in the model has a name, a priority level, a duration probability, which is distributions with respect to possible team association, and a set of pre-requisite tasks. The proxel format of the state of the project schedule encompasses the following parameters: task vector [47], where (Ti,Tj) reflects the fact that Team A is working on Ti and Team B is working on Tj respectively, the age intensity vector {τi}, for tracking the duration of tasks, probability value and the probability for the model being the denoted state. I is used to reflect an idle team.

Thus the format of the proxel is as follows:

Proxel = (Task Vector, Age Intensity Vector, Probability)

The initial proxel, i.e. the proxel that marks the initial state of the system would be ((T1, T3), (0, 0), 1.0). It describes the situation in which *Team A* is working on task T1, and *Team B* on task T3 with a probability of 1.0. In the next time step the model can do each of the following developments:

- Task T1 is completed,
- Task T3 is completed, or
- None of the tasks are completed

Resulting into the following three proxels:

a) ((T2, T3), (0, Δt), p1)

b) ((T1, T4), (Δt, 0), p2)

c) ((T1, T5), (Δt, 0), 1 - p1 - p2)

In case (a), team A starts working on task T2, and also the corresponding age intensity is now reset to track the duration of T2. Case (b) and (c) show the cases when team B finishes Task 3, it will either start T4 or T5. We recall that team B cannot start T6 because of the precedence constraints with T1 which in turn will minimize the idle time of teams. Team B cannot start T8 because T4 and T5 have higher priority than T8.

During the state exploration, around 40 discrete states have been generated. Two facts contributed in doubling the number of states that can be reached: 1) no sequencing is followed except the respect of the ordering constraints, and 2) the introducing of three floating tasks that can be implemented by Team A or Team B according to their availability.

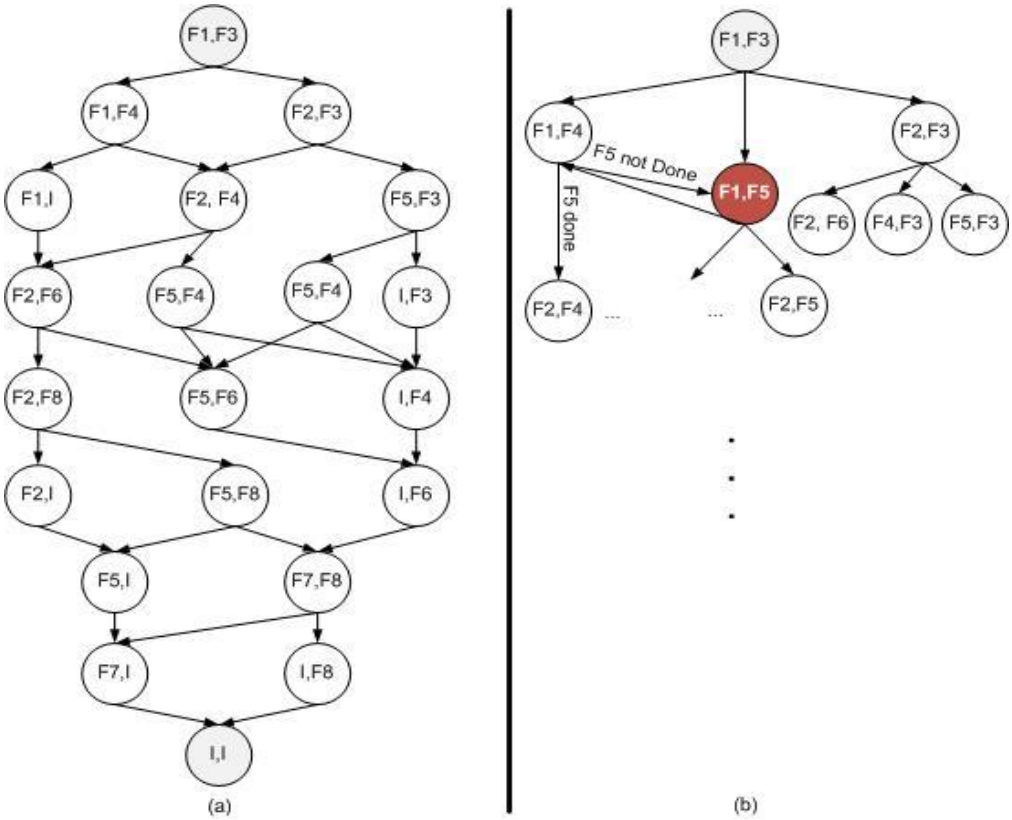

Fig. 6. State space of the HOLIS system.

6.3. Simulation with Priority Consideration and Fuzzy Rules (Case C)

In this simulation model, we allow the cancellation of tasks by adding fuzzy rules. T7 and T8 in this case are considered as cancellable tasks. Adding fuzzy rules will add few states during the state exploration comparing the previous model. However, it will add more connections to the states of the state exploration graph. As an example, let us consider the state ((T2, T4), (0, 0), 1.0). It describes the situation in which Team A is working on task T2, and Team B on task T4 with a probability of 1.0 and where Tasks T1, T3, T6,T5 are already implemented. In the next time step the model can do each of the following developments:

- Task T2 is completed,
- Task T4 is completed, or
- None of the tasks are completed

Normally, this will result into the following four proxels:

a) ((T7, T4), (0, Δt), p1)
b) ((T2, T7), (Δt, 0), p2×0.5)
c) ((T2, T8), (Δt, 0), p2×0.5)
d) ((T2,T4),(Δt ,Δt),1 - p1 - p2)

Proxels (b) and (c) show the probability for TeamB to start working on T7 and T8 after implementing Feature4 since both tasks are representing useful features.

Making T7 and T8 cancellable change this fact and proxel ((T2, T4), (0, 0), 1.0) will lead to the following six proxels:

a) ((T7, T4), (0, Δt), p1× (1-pcancel))
b) ((N, T4), (0, Δt), p1×pcancel)
c) ((T2, T7), (Δt, 0), 0.5× p2 × (1-pcancel))
d) ((T2, T8), (Δt, 0), 0.5 × p2 × (1-pcancel))

e) ((T2, N), (Δt, 0), p2×pcancel)

f) ((T2, T4), (Δt, 0), 1 - p1 - p2)

Pcancel represents the probability to cancel the implementation of a cancellable task. Consequently, Team A can start implementing T7, but can also stay idle if the project is close to deadline. Same applies for team B working on T7 and T8.

6.4. Simulation Result and Discussion

In order to evaluate our proposed model, we discuss simulation results based on three criteria: probability to complete the project on time, probability to complete all tasks by the end of the project and finally the probabilities to finish each task individually by the deadline.

6.4.1. Project completion by deadline

Fig. 7 shows the simulation results for the three cases. The deadline to complete the project was set at 20 time units. The simulation shows that following the classical schedule, where priority of features, as well as, flexibility in human resources allocation is not considered, the project will never end on time while following the second model (Case B) , where no fuzzy remedial rules are defined, a small improvement is observed but still the project will not be delivered within the deadline. However, this probability reached 60% in the case of the EPS model. In addition, even by extending the deadline to 30 time units, while the probabilities to finish the project increase but still the observations do not change. As expected, the classical schedule (Case A) yields a probability to complete the project as 75%. In the second model (Case B) simulation shows that the probability to finish the project by the deadline is approximately 85%. The highest project completion probability is for the EPS model, where the probability to finish the project by the deadline is higher than 95%.

This result confirms that:

1) Rigid human allocation of tasks does not optimize the usage of human resources. Even without cancelling the implementation of useful features, the difference in probability between the first and the second model is roughly 10% which represents a considerable difference for the project manager. This is attributed to the floating tasks.
2) Implementing the cancelation of task in EPS enhances the schedule accuracy and adaptability. It allows the manager to assess the risk and negotiate with customers about the possibility of task implementation cancelation if the project runs out of budget and /or time. This would save money and time, and focus the project work on the critical tasks.
3) In this particular project, cancellation of useful features under certain circumstances (being close to the deadline) to meet the deadline seems to be a good option.

Obviously, more parameters need to be included, such as cost, to make a complete decision. However, the simulation results can cut a significant amount of the risk in making these decisions.

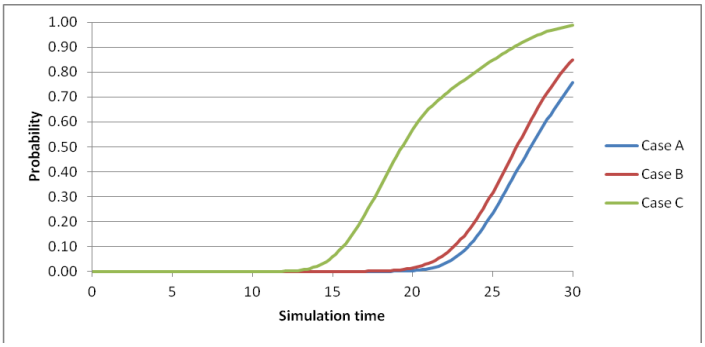

Fig. 7. Probability of completing the project.

6.4.2. Task completed by model

Fig. 8 shows the probability to implement all tasks, as specified in the initial project schedule model. Case B model has the highest probability to have all tasks implemented by the deadline, while EPS and the classic schedule converge to having very close values (almost identical) at the deadline. As time goes by, beyond the deadline, this is changing, ultimately yielding the highest probability to Case A, and the lowest to Case C, which would be the expected. However, the important cutting point is the deadline, and at that time we can see no significant difference in the probability values that would oppose the scenario described by the Case C. This would form part of the assessment of the possible decision that the manager can make to meet the project deadline.

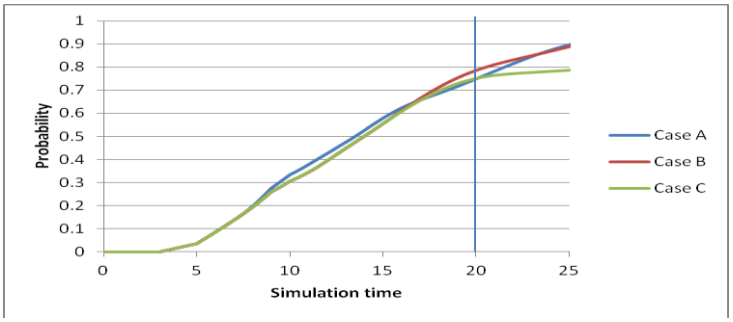

Fig. 8. Probability of having all tasks completed.

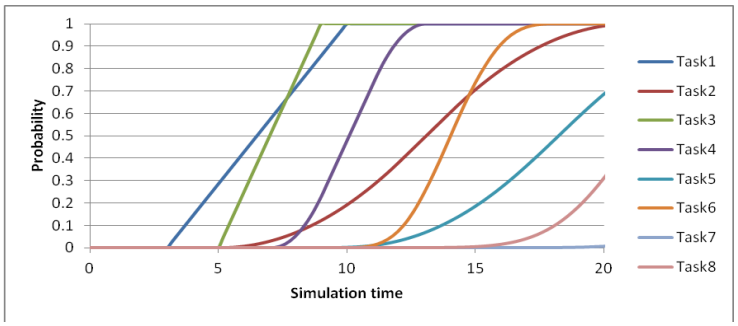

Fig. 9. Probability of individual task completion with classical schedules.

6.4.3. Implementation of individual tasks per model

The third interesting aspect to simulate is the probability of implementing each individual task per model. To study this aspect, we will consider the cases of critical, important and nice-to-have features. The simulation results for the three models (Case A, Case B, and Case C) are shown in Fig. 9, Fig. 10 and Fig. 11, respectively.

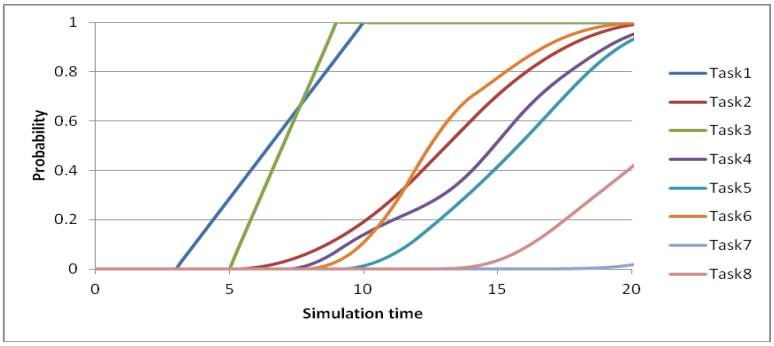

Fig. 10. Probability of individual task completion with schedules with priorities and floating tasks.

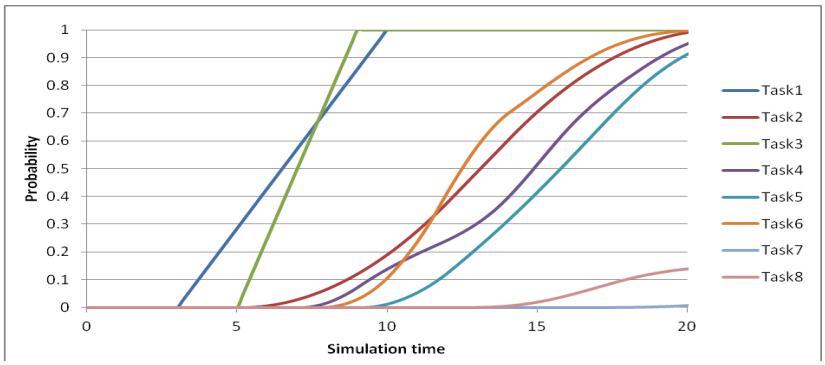

Fig. 11. Probability of individual task completion with EPS.

Comparing the simulation results leads to three main conclusions:

1) In the classical schedule, the static sequencing of tasks may delay the implementation of critical/important tasks. As we can observe, the implementation of Task 6 will not start before time unit 10 in the first schedule, whereas it will start around time unit 8 in the second and the third schedule.
2) Task 5 has a very low probability of implementation by the project deadline in the case of the classical schedule (Case A). This issue is not observed in the second and the third cases because Task 5 has more priority than Task 7 and Task 8, a fact that is taking in consideration while generating the state space.
3) The EPS shows the highest probability to implement each critical and important task. It also has the lowest probability to implement the two useful features represented by Task 7 and Task 8.

Finally, comparing the results of the second and the third case, the manager may consider to change the RAS and to re-simulate the EPS by considering the cancellation of Task 7 only. Our model is proposed as a decision-making assistant as well as an approach to assess various decisions that managers may take in order to handle uncertainties. As such, it will certainly guide managers in the typical wealth of information that they are surrounded by.

7. Conclusion and Future Work

This paper presents a methodology that uses more realistic and robust project schedule modeling in simulating project plans that allows for a high level of uncertainty in resource constraints. This model uses the different priorities of software features to adequately represent them in the simulation model. For that, a new resource-constrained project scheduling in which tasks are classified in four categories according to the nature and level of importance of the features they are implementing is presented.

The methodology the paper is proposing incorporates EPS in project management. It allows the generation of schedules that: (a) handles the uncertainty in human resources allocation to different project tasks and gives flexibility in teams' allocation, (b) simulates various on-the-fly human decisions and their impact on the project duration, and finally (c) handles effectively delays that may occur while implementing different features. Ultimately, schedules produced using this methodology are more robust since they are less sensitive to resources unavailability: a delay of one activity would not necessary lead to a schedule disruption. By comparing different models, we proved that EPS is in fact more informative to the manager since it allows him to simulate different possible remedial scenarios and to make decision on what scenario to adopt with regards to the project goal. For instance, it would be simpler for managers to predict ahead of time which useful feature will be implemented and those that are not within the given project time frame. Following an iterative process, a manager may refine his features selection and re-build the corresponding simulation model. This will help deciding on the nice-to-have features to keep and those to eliminate,

helping consequently the process of negotiation with the customer.

To extend our work we plan to address the effect of these uncertainty factors on the productivity and budget, by adding value, effort and cost parameters. In addition, we plan to enrich our simulation model to measure the effect of features volatility that software projects are suffering from. To handle features volatility, the model needs to be extended with task interruption.

References

[1] Herroelen, W. (2005). Project scheduling-Theory and practice. *Production and Operations Management*, *14*, 413-432.

[2] Neimat, T. A. (2014). Why IT Projects Fail?. Retrieved 2014, from http://www.projectperfect.com.au/info_it_projects_fail.php

[3] Pinto, J. K. (2002). Project management.

[4] Belassi, W., & Tukel, O. I. (1996). A new framework for determining critical success/failure factors in projects. *International Journal of Project Management*, *14*, 141-151.

[5] Charette, R. N. (2005). Why Software Fails?. Retrieved 2005, from http://spectrum.ieee.org/computing/software/why-software-fails/

[6] Wen, Z. J., & Fang, S. H. (2009). Multi-mode double resource-Constrained time/cost trade-offs project scheduling problems. *Proceedings of the International Conference on Management and Service Science* (pp. 1-4).

[7] Huang, W., Ding, L., Wen, B., & Cao, B. (2009). Project scheduling problem for software development with random fuzzy activity duration times.

[8] Herroelen, W., & Leus, R. (2005). Project scheduling under uncertainty: Survey and research potentials. *European Journal of Operational Research*, *165*, 289-306.

[9] Arauzo, J. A., Galán, J. M., Pajares, J., & Paredes, A. L. (2009). Multi-agent technology for scheduling and control projects in multi-project environments. An Auction based approach. *Inteligencia Artificial*, *42*, 12-20.

[10] Sobel, M. J., Szmerekovsky, J. G., & Tilson, V. (2009). Scheduling projects with stochastic activity duration to maximize expected net present value. *European Journal of Operational Research*, *198*, 697-705.

[11] Perminova, O., Gustafsson, M., & Wikström, K. (2008). Defining uncertainty in projects — A new perspective. *International Journal of Project Management*, *26*, 73-79.

[12] Port, D., Olkov, A., & Menzies, T. (2008). Using simulation to investigate requirements prioritization strategies. *Proceedings of the 2008 23rd IEEE/ACM International Conference on Automated Software Engineering*.

[13] Mizouni, R., & Molnar, S. L. (2012). Simulation-based feature selection for software requirements baseline. *Journal of Software*.

[14] Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B., & Bush, M. (1993). K*ey Practices of the Capability Maturity Model, Version 1.1. Tech. Rept. CMU/SEI-93-TR-25*. Software Engineering Institute, Carnegie Mellon University Pittsburgh, Pennysylvania Tech. Rept. CMU/SEI-93-TR-25, 1993.

[15] Firesmith, D. (2004). Prioritizing requirements. *Journal of Technology*, *3*, 35-48.

[16] Leffingwell, D., & Widrig, D. (2003). M*anaging Software Requirements: A Use Case Approach.* Pearson Education.

[17] Pfahl, D., & Lebsanft, K. (2000). Using simulation to analyse the impact of software requirement volatility on project performance. *Information and Software Technology*, *42*, 1001-1008.

[18] Olivier, L., Erik, D., & Willy, H. (2008). Proactive and reactive strategies for resource-constrained project

scheduling with uncertain resource availabilities. *Journal of Scheduling*, *11*, 121-136.

[19] Ferreira, S., Shunk, D., Collofello, J., Mackulak, G., & Dueck, A. (2011). Reducing the risk of requirements volatility: findings from an empirical survey. *Journal of Software Maintenance and Evolution: Research and Practice*, *23*, 375-393.

[20] Mizouni, R., & Salah, A. (2010). Towards a framework for estimating system NFRs on behavioral models. *Knowledge-Based Systems*, *23*, 721-731.

[21] S. Ferreira, J. Collofello, D. Shunk, & Mackulak, G. (2009). Understanding the effects of requirements volatility in software engineering by using analytical modeling and software process simulation. *Journal of Systems and Software*, *82*, 1568-1577.

[22] Leus, R. (2004). The generation of stable project plans. *A Quarterly Journal of Operations Research*, *2*, 251-254.

[23] Daneva, M. (2010). Balancing uncertainty of context in ERP project estimation: an approach and a case study. *Journal of Software Maintenance and Evolution: Research and Practice*, *22*, 329-357.

[24] Vonder, S., Demeulemeester, E., Leus, R., & Herroelen, W. (2006). Proactive-reactive project scheduling trade-offs and procedures. *Perspectives in Modern Project Scheduling*, 25-51.

[25] Sadeh, N., Otsuka, S., & Schelback, R. (1993). Predictive and reactive scheduling with the microboss production scheduling and control system. *Proceedings of the IJCAI-93, Workshop on Knowledge-Based Production Planning, Scheduling and Control* (pp. 293-306).

[26] Sakkout, H. E., & Wallace ,M. (2000). Probe backtrack search for minimal perturbation in dynamic scheduling. *Constraints*, *5*, 359-388.

[27] Alagöz, O., & Azizoglu, M. (2003). Rescheduling of identical parallel machines under machine eligibility constraints. *European Journal of Operational Research*, *149*, 523-532.

[28] Stork, F. (2001). *Stochastic Resource-Constrained Project Scheduling*. Technische Universitat Berlin.

[29] Prade, H. (1979). Using fuzzy set theory in scheduling problem: A case study. *Fuzzy Sets and Systems*, *2*, 153-165.

[30] Soltani, A., & Haji, R. (2007). A project scheduling method based on fuzzy theory. *Journal of Industrial and Systems Engineering*, *1*, 70-80.

[31] Tse, L. F., & Y. S. Jing. (2003). Fuzzy critical path method based on signed-distance ranking and statistical confidence-interval estimates. *Journal Supercomput*, *24*, 305-325.

[32] Oliveros, A. V. O., & Fayek, A. R. (2005). Fuzzy Logic approach for activity delay analysis and schedule updating. *Journal of Construction Engineering and Management*, *131*, 42-51.

[33] Herroelen, W., & Leus, R. (2004). The construction of stable project baseline schedules. *European Journal of Operational Research*, *156*, 550-565.

[34] Penz, B., Rapine, C., & Trystram, D. (2001). Sensitivity analysis of scheduling algorithms. *European Journal of Operational Research*, *134*, 606-615.

[35] Kellner, M. I., Madachy, R. J., & Raffo, D. M. (1999). Software process simulation modeling: Why? What? How? *Journal of Systems and Software*, *46*, 91-105.

[36] Barachi, M. E., Glitho, R., & Dssouli, R. (2011). Control-level call differentiation in IMS-based 3G core networks. *Network*, *25*, 20-28.

[37] Wahab, O. A., Otrok, H., & Mourad, A. (2013). VANET QoS-OLSR: QoS-based clustering protocol for vehicular ad hoc networks. *Computer Communications*, *36*, 1422-1435.

[38] Cayirci, E. (2013). Modeling and simulation as a cloud service: A survey. *Proceedings of the 2013 Winter Simulation Conference: Simulation: Making Decisions in a Complex World*.

[39] Lazarova, S. M., & Mizouni, R. (2010). Modeling human decision behaviors for accurate prediction of project schedule duration. Hammamet, Tunisia.

[40] Horton, G. (2002). A new paradigm for the numerical simulation of stochastic Petri nets with general firing times. *Proceedings of the European Simulation Symposium*.

[41] Isensee, C., Molnar, S. L., & Horton, G. (2005). Combining proxels and discrete phases. *Proceedings of International Conference on Modeling, Simulation and Applied Optimization*.

[42] Stewart, W. J. (1994). *Introduction to the Numerical Solution of Markov Chains*, Princeton University Press.

[43] Cox, D. R. (1955). The analysis of non-markovian stochastic processes by the inclusion of supplementary variables. *Proceedings of the Cambridge Philosophical Society* (pp. 433-441).

[44] Lin, F. J., Chu, P. M., & Liu, M. T. (1987). Protocol verification using reachability analysis: The state space explosion problem and relief strategies. *Computer Communication Review*, *17*, 126-135.

[45] Isensee, C., & Horton, G. (2005). Approximation of discrete phase-type distributions. *Proceedings of the 38th annual Symposium on Simulation* (pp. 99-106).

[46] Li, C., Akker, M. V. D., Brinkkemper, S., & Diepen, G. (2010). An integrated approach for requirement selection and scheduling in software release planning. *Requirements Engineering*, *15*, 375-396.

[47] AbouRizk, S. M., & Wales, R. J. (1997). Combined discrete-event/continuous simulation for project planning. *Journal of Construction Engineering and Management*, *123*, 11-20.

Rabeb Mizouni is an assistant professor in software engineering at Khalifa University. She got her PhD and her MSc in electrical and computer engineering from Concordia University, Montreal, Canada in 2007 and 2002 respectively. Previously, she was appointed as an assistant professor at the College of Information Technology at the United Arab Emirates University. Currently, she is interested in the deployment of context aware mobile applications, software product line and cloud computing.

Sanja Lazarova Molnar is currently working as an associate professor at the center for energy informatics, under the Faculty of Engineering, at the University of Southern Denmark. Previously, she was appointed as an assistant professor at the College of InformationTechnology at the United Arab Emirates University. Sanja obtained her Ph.D. in computer science from the University "Otto-Von-Guericke" in Magdeburg, Germany where she was also a member of the Simulation Group at the Institute of Simulation and Graphics